

**DISK DRIVE USING ROTATIONAL POSITION OPTIMIZATION ALGORITHM TO  
FACILITATE WRITE VERIFY OPERATIONS**

**BACKGROUND OF THE INVENTION**

**Field of the Invention**

The present invention relates to disk drives. More particularly, the present invention relates to a disk drive using a rotational position optimization (RPO) algorithm to facilitate write verify operations.

**Description of the Prior Art**

It has been suggested to verify write operations in a disk drive by reading recently written data sectors to verify their recoverability before releasing the cache memory storing the write data. If a recently written data sector cannot be recovered, the write data stored in the cache memory is rewritten to the disk. In effect, the write verify operation detects and compensates for "bad writes" due, for example, to an abnormal fly height or other random anomaly not detected during the write operation. Due to the rotational latency needed to reread a data sector as well as the rotational latency needed to rewrite the data sector, the prior art has suggested various methods for performing the write verify operation so as to minimize the impact on performance.

U.S. Patent No. 5,872,800 suggests to perform a write verify operation "off-line" while the disk drive is in an idle mode and not processing commands received from the host computer. In this manner, the write verify operation does not impact the performance of the disk drive while processing host commands during normal operation. However, this technique requires a significant amount of cache memory in order to cache the write data for all of the write commands received from the host while the disk drive is "on-line", as well as the read data associated with read commands received during the same "on-line" period. If the cache memory is exhausted during an "on-line" period, at least some of the write verify commands will be discarded which is undesirable. This problem is exacerbated if the "off-line" periods are short due to frequent access to the disk drive by the host computer.

1 U.S. Patent No. 6,289,484 discloses a write verify technique wherein the disk drive  
2 performs an "off-line" scan of all previously written data sectors. If during the off-line scan a data  
3 sector cannot be recovered on-the-fly using the sector level redundancy, then the data sector is  
4 added to a "bad block" list. If during an "on-line" mode the disk drive attempts to write to a data  
5 sector in the bad block list, the disk drive will perform an immediate write verify operation on the  
6 data sector. In this manner, only the suspect data sectors in the bad block list are verified after a  
7 write operation in order to minimize the impact on performance. However, a bad write can occur  
8 due to a random anomaly, such as an abnormal fly height, which may not be detected during the  
9 off-line scan.

10 U.S. Patent No. 5,588,007 discloses a write verify technique wherein the disk drive  
11 monitors the on-line write operations for abnormal fly height conditions. If an abnormal fly height  
12 condition is detected during a write operation, the disk drive performs an immediate write verify  
13 operation on the data sector. However, a bad write may occur due to a random anomaly other  
14 than an abnormal fly height.

15 There is, therefore, the need to improve the write verify operation of a disk drive by  
16 minimizing the cache memory requirements while maximizing the probability that every  
17 undetected bad write is corrected.

## 18 SUMMARY OF THE INVENTION

19 The present invention may be regarded as a disk drive comprising a disk having a plurality  
20 of concentric tracks, each track comprising a plurality of data sectors, and a head actuated radially  
21 over the disk. The disk drive further comprises an input/output (I/O) queue for storing read and  
22 write commands received from a host computer, and a disk controller for executing the  
23 commands stored in the I/O queue in an order determined from a rotational positioning  
24 optimization (RPO) algorithm. The disk controller selects a write command from the I/O queue  
25 according to the RPO algorithm, seeks the head to a target track, and writes data to a target data  
26 sector. After executing the write command, the disk controller inserts a write verify command  
27 into the I/O queue. The disk controller then selects the write verify command from the I/O queue

1 according to the RPO algorithm and executes the write verify command to verify the  
2 recoverability of the data written to the target data sector.

3 In one embodiment, the disk drive comprises a semiconductor memory comprising a  
4 plurality of blocks for storing write data received from the host computer and for storing read  
5 data read from the disk. The block of semiconductor memory storing the write data for the target  
6 data sector is de-allocated after the disk controller executes the write verify command. In another  
7 embodiment, the block of semiconductor memory storing the write data for the target data sector  
8 is de-allocated prior to the disk controller executing the write verify command if the amount of  
9 free blocks in the semiconductor memory is less than a predetermined threshold. In one  
10 embodiment, the I/O queue stores a plurality of write verify commands corresponding to a  
11 plurality of written data sectors, and when the number of free blocks in the semiconductor  
12 memory is less than the predetermined threshold, the disk controller uses a predetermined criteria  
13 to delete at least one of the write verify commands from the I/O queue and to de-allocate the  
14 corresponding block of semiconductor memory. In one embodiment, the predetermined criteria  
15 deletes the write verify command that optimizes the RPO algorithm with respect to the remaining  
16 commands in the I/O queue. In another embodiment, each write command comprises data to be  
17 written to one or more data sectors, and the predetermined criteria deletes the write verify  
18 command comprising the least number of data sectors to be verified compared to other write  
19 verify commands in the I/O queue.

20 In one embodiment, the disk controller executes the write verify command by seeking the  
21 head to the target track and reading data from the target data sector. If the read fails, the disk  
22 controller selects the write command from the I/O queue according to the RPO algorithm and  
23 executes the write command. In one embodiment if the read fails and the write data associated  
24 with the write command has been de-allocated prior to the disk controller executing the write  
25 verify command, the disk controller executes a firmware error-recovery procedure to recover the  
26 write data stored in the target data sector, stores the recovered write data in a semiconductor  
27 memory, and inserts a write command into the I/O queue for writing the write data to the target

data sector. In one embodiment if a write verify fails multiple times for the same write command, the disk controller inserts a relocate command into the I/O queue for processing according to the RPO algorithm, the relocate command for relocating an errant data sector.

The present invention may also be regarded as a method of executing a write verify operation in a disk drive, the disk drive comprising a disk having a plurality of concentric tracks including a plurality of data sectors, a head actuated radially over the disk, and an input/output (I/O) queue for storing read and write commands received from a host computer. A write command is selected from the I/O queue according to a rotational positioning optimization (RPO) algorithm. After seeking the head to a target track, data is written to a target data sector. A write verify command is inserted into the I/O queue, and then selected from the I/O queue according to the RPO algorithm. The write verify command is executed to verify the recoverability of the data written to the target data sector.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A and 1B show a disk drive according to an embodiment of the present invention wherein write verify commands are executed according to a rotational positioning optimization (RPO) algorithm.

FIG. 2 shows a disk drive according to an embodiment of the present invention comprising a semiconductor memory comprising an I/O queue for storing read, write and write verify commands, and a cache memory for storing user data associated with the commands.

FIG. 3 is a flow chart according to an embodiment of the present invention illustrating how resources are freed to facilitate a recently received write command.

FIG. 4 is a flow chart according to an embodiment of the present invention illustrating how read, write, and write verify commands are executed according to the RPO algorithm wherein write verify commands are deleted from the I/O queue when resources are needed to process new commands.

FIG. 5 is a flow chart according to an embodiment of the present invention illustrating how read, write, and write verify commands are executed according to the RPO algorithm

wherein cache memory for pending write verify commands is de-allocated when resources are needed to process new commands without deleting the pending write verify commands from the I/O queue.

#### DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1A shows a disk drive 2 comprising a disk 4 having a plurality of concentric tracks 6, each track comprising a plurality of data sectors, and a head 8 actuated radially over the disk 4. The disk drive 2 further comprises an input/output (I/O) queue 10 for storing read and write commands received from a host computer, and a disk controller 12 for executing the commands stored in the I/O queue 10 in an order determined from a rotational positioning optimization (RPO) algorithm. FIG. 1B is a flow diagram illustrating the steps executed by the disk controller 12 when performing a write operation. At step 14 the disk controller 12 selects a write command from the I/O queue 10 according to the RPO algorithm, and at step 16 seeks the head 8 to a target track and writes data to a target data sector. After executing the write command, at step 18 the disk controller 12 inserts a write verify command into the I/O queue 10. At step 20 the disk controller 12 then selects the write verify command from the I/O queue 10 according to the RPO algorithm and at step 22 executes the write verify command to verify the recoverability of the data written to the target data sector.

In the embodiment of FIG. 1A, the disk 4 further comprises a plurality of embedded servo sectors 24 which are processed by the disk controller 12 to servo control the radial location of the head 8. Each embedded servo sector 24 comprises coarse head positioning information (e.g., a track number) as well as fine positioning information in the form of servo bursts recorded at precise offsets from the target track's centerline.

Any suitable RPO algorithm may be employed in the present invention. In general, the RPO algorithm computes an estimated access time for each command in the I/O queue, taking into account the current radial and circumferential location of the head. The RPO algorithm selects the command having the smallest access time as the next command to execute. This enhances performance of the disk drive by maximizing efficiency relative to its mechanical

latencies. The performance of the disk drive 2 in FIG. 1A is further enhanced by executing write verify commands in an optimal manner as determined from the RPO algorithm. In some cases it may be that a write verify command is the next best command to execute after a write command. However, there are cases where it is more efficient to perform at least one other command (e.g., another write command to the same or proximate track) before executing a write verify command for a previous write operation.

In one embodiment, the disk controller 12 executes the write verify command by seeking the head 8 to the target track and reading data from the target data sector. If the read fails, the disk controller 12 selects the write command from the I/O queue 10 according to the RPO algorithm and re-executes the write command. In one embodiment, a write command is deleted immediately from the I/O queue 10 after performing the write command. If the corresponding write verify command fails, the write command is re-inserted into the I/O queue 10. In an alternative embodiment, the write command is not deleted from the I/O queue 10 until the write verify command is successfully executed. In this manner it is not necessary to re-insert the write command into the I/O queue if the corresponding write verify command fails. In yet another embodiment, the original write command is modified into a write verify command after performing the write operation. If the write verify fails, the write verify command is modified back into a write command in order to re-execute the write command. If the write verify fails a number of times which exceeds a predetermined threshold, the write verify command is modified into a relocate command in order to relocate an errant data sector(s).

In an embodiment shown in FIG. 2, the disk drive 2 comprises a semiconductor memory 26 which may be volatile (e.g. DRAM) or non-volatile (e.g., FLASH). The semiconductor memory 26 stores the I/O queue 10 which may be implemented in any suitable manner, such as an array, a linked list of pointers, or multiple queues which collectively form the I/O queue. The semiconductor memory 26 is also used to implement a cache memory 28 for caching user data associated with the read and write commands received from the host. When the disk drive 2 receives a write command from the host, the user data to be written to the disk is cached in the

cache memory 28 until the write command is successfully executed and verified. In one embodiment, if the amount of free memory in the cache memory 28 decreases to a predetermined limit, the memory allocated for a pending write verify command is de-allocated in order to free memory for subsequent commands. In one embodiment, the pending write verify command is deleted, and in another embodiment, the pending write verify command is still executed even though the memory storing the write data has been de-allocated.

FIG. 3 shows a flow diagram executed by the disk controller 12 for processing a write command received from a host, including de-allocating memory for pending write verify commands if necessary. At step 30 the disk controller 12 receives a write command from the host and at step 32 the disk controller 12 determines whether there are sufficient resources (including cache memory 28) to service the write command. If resources are needed, then at step 34 the disk controller 12 frees resources associated with unneeded commands, for example, by de-allocating cache memory 28 associated with the least recently requested read or write commands that have been successfully executed but which still have cached user data.

In one embodiment, at step 34 the disk controller 12 de-allocates cache memory 28 for one or more pending write verify commands to free memory for the write command received from the host at step 30. In one embodiment, the pending write verify commands are deleted from the I/O queue 10, and in an embodiment described below with reference to FIG. 5, the write verify commands are executed without the write data being cached. In one embodiment, the disk controller 12 uses a predetermined criteria to delete at least one of the write verify commands from the I/O queue 10. In one embodiment, the predetermined criteria deletes the write verify command that optimizes the RPO algorithm with respect to the remaining commands in the I/O queue 10. In another embodiment, each write command comprises data to be written to one or more data sectors and the predetermined criteria deletes the write verify command comprising the least number of data sectors to be verified compared to other write verify commands in the I/O queue 10.

If after deleting the unneeded commands at step 36 more resources are still needed and at

1 step 38 there are no more unneeded commands that can be deleted, then at step 40 the disk  
2 controller 12 flushes dirty write commands according to the RPO algorithm. A dirty write  
3 command is a write command which is pending execution and therefore has user data stored in  
4 the cache memory. In effect, pending write commands are expedited at step 40 by assigning a  
5 higher priority in the RPO algorithm. The dirty writes are flushed at step 40 until sufficient  
6 resources are available at step 42 to process the new write command received from the host at  
7 step 30. At step 44 the user data associated with the new write command is stored in the cache  
8 memory 28, at step 46 the head/cylinder/wedge numbers are calculated for the new write  
9 command, and at step 48 the new write command is placed in the I/O queue 10 for processing  
10 according to the normal RPO algorithm.

11 Referring to the flow diagram of FIG. 4, at step 50 the disk controller 12 selects the next  
12 command to execute from the I/O queue 10 according to the RPO algorithm. If at step 52 the  
13 next command to execute is the write command received at step 30 (FIG. 2), then at step 54 the  
14 disk controller 12 executes the write command and inserts a corresponding write verify read  
15 command in the I/O queue at step 56. Control then branches back to step 50 to select the next  
16 command to execute from the I/O queue 10.

17 If at step 52 the next command to execute is a read command, then at step 58 the disk  
18 controller 12 executes the read command. If at step 60 the read is successful and at step 62 the  
19 read command was a write verify read command, then at step 64 the cache memory 28 storing the  
20 write data for the verified write command is de-allocated. If a read error occurs at step 60 (e.g.,  
21 unable to recover a data sector using on-the-fly ECC) and at step 61 the read command was a  
22 write verify read command, then at step 63 a branch is executed based on the number of times the  
23 write verify command has been attempted. If the number has not exceeded a predetermined  
24 threshold, then at step 64 a write command is inserted into the I/O queue 10 so that the write  
25 command will be re-executed by again writing the write data to the target data sectors. The write  
26 command is re-executed according to the RPO algorithm when selected from the I/O queue at  
27 step 50.

1 If at step 63 a write verify command has failed a number times that exceeds the  
2 predetermined threshold, then at least one of the target data sectors responsible for the read error  
3 is relocated by inserting a relocate command into the I/O queue 10 at step 65. Relocating a data  
4 sector involves re-mapping a logical block address from the physical block address of an errant  
5 data sector to the physical block address of a spare data sector. The write data stored in the  
6 cache 28 is then written to the spare data sector. In order to optimize performance, the write  
7 command for writing the write data to the spare data sector is executed according to the RPO  
8 algorithm. In one embodiment, the data written to the spare data sector is verified and relocated  
9 yet again if necessary.

10 If at step 61 the read error occurred for a normal read command received from the host,  
11 then at step 66 a branch is executed according to a number of retry reads attempted on the errant  
12 data sector. If a number of retry reads has not exceeded a predetermined threshold, then at step  
13 68 a retry read is attempted.. If at step 66 the number of retry reads has exceeded the  
14 predetermined threshold, then at step 70 the disk controller 12 executes a number of heroic  
15 firmware error recovery procedures in an attempt to recover the errant data sector. The heroic  
16 firmware procedures may include adjusting various read channel parameters such as the equalizer  
17 filter coefficients, introduce an offset into the servo tracking algorithm, or adjusting other  
18 parameters which may enable successful recovery of the data sector. If the heroic firmware  
19 procedures are successful at step 72, then at step 74 a write command is inserted into the I/O  
20 queue 10 so that the errant data sector will be re-written and re-verified. If the heroic firmware  
21 procedures are not successful at step 72, then at step 76 an error is returned to the host computer  
22 indicating that the read operation failed.

23 Referring again to FIG. 3, in one embodiment at step 34 the cache memory 28 for selected  
24 write verify commands is de-allocated without actually deleting the write verify command from  
25 the I/O queue 10. The write verify command is still executed according to the RPO algorithm,  
26 and if the write verify fails, the heroic firmware procedures are executed in an attempt to recover  
27 data from an errant data sector. This embodiment is understood with reference to the flow

1 diagram of FIG. 5. If at step 78 the write data is still cached after a write verify read has failed,  
2 then control proceeds to step 63 similar to FIG. 4. However, if the cache memory 28 for storing  
3 the write data was de-allocated at step 34 of FIG. 3, then control proceeds to step 66. If at step  
4 72 the heroic firmware procedures are not successful in recovering the errant data sector, a  
5 branch is executed at step 80. If at step 80 the failed read operation was for a write verify  
6 command, then control proceeds at step 50 effectively ignoring the error since a new write  
7 command may over write the unrecoverable data sector. Otherwise, an error will be returned to  
8 the host at step 76 if a normal read operation attempts to read the errant data sector.

9 The disk controller 12 comprises suitable circuitry and/or software for processing the  
10 commands received from the host, such as processor circuitry, interface circuitry, an error  
11 correction code (ECC), a read/write channel, servo control, etc.. The disk controller 12 may be  
12 implemented as a plurality of distinct integrated circuits, or as a single integrated circuit.